SiRen: Leveraging Similar Regions for Efficient & Accurate Variant Calling

Kristal Curtis
Ameet Talwalkar
Matei Zaharia
Armando Fox
David A. Patterson

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-159
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-159.html

May 30, 2015

Copyright © 2015, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Acknowledgement

The authors thank Jesse Liptrap and Julie Newcomb for help with the variant calling experiments. They also thank the AMP-X team at UC Berkeley for helpful discussions and Beth Trushkowsky for feedback on the draft.
This research is supported in part by NSF CISE Expeditions Award CCF-1139158, LBNL Award 7076018, and DARPA XData Award FA8750-12-2-0331, and gifts from Amazon Web Services, Google, SAP, The Thomas and Stacey Siebel Foundation, Adatao, Adobe, Apple, Inc., Blue Goji, Bosch, C3Energy, Cisco, Cray, Cloudera, EMC2, Ericsson, Facebook, Guavus, HP, Huawei, Informatica, Intel, Microsoft, NetApp, Pivotal, Samsung, Schlumberger, Splunk, Virdata and VMware.

SiRen: Leveraging **Si**milar **Re**gio**n**s for Efficient & Accurate Variant Calling

Kristal Curtis,[1] Ameet Talwalkar,[2,4] Matei Zaharia,[3,4]
Armando Fox,[1] and David A. Patterson[1]

[1]Department of EECS, University of California, Berkeley, CA 94720

[2]Computer Science Department, UCLA, Los Angeles, CA, 90095

[3]Department of EECS, Massachusetts Institute of Technology, Cambridge, MA 02139

[4]Databricks Inc., San Francisco, CA 94105

Abstract

Next-generation genomic sequencing costs are rapidly decreasing, having recently reached the $1000-per-genome barrier, a likely tipping point for widespread clinical use. However, genomic analysis techniques have failed to keep pace. In particular, the process of variant calling, or inferring a sample genome from the noisy sequencing data, introduces major computational and statistical challenges. In this work, we explore the feasibility of a hybrid approach that addresses these challenges by partitioning the genome into easier and harder regions, deploying efficient algorithms on the easier regions, and relying on more expensive and accurate technologies in the harder regions. We propose that near duplication, or similarity, in the genome is a natural signal for identifying harder regions, and then present a large-scale distributed clustering approach to identify these similar regions.

We perform an extensive empirical study illustrating the effectiveness of existing variant calling algorithms on the easier regions and their contrasting struggles on the similar regions. We also confirm that the *similar* regions are sufficiently disjoint, thus providing the opportunity for sophisticated analysis of these regions in an embarrassingly parallel manner.

1 Introduction

Advances in DNA sequencing technologies have the potential to change the landscape of biological research [17]. Over a decade ago, the Human Genome Project [9] sequenced the human genome on a $3B budget. However, as a result of massively parallel sequencing machines which produce *short reads*, or DNA segments of around 100 characters, sequencing costs are decreasing at a rate faster than Moore's Law, and will soon reach the $1000 per genome barrier. This technology has the potential for widespread clinical usage in the near future, pending the development of short-read analysis pipelines of sufficient accuracy and efficiency.

Prior to semantic analysis, such as whether a person is likely to get cancer, short reads must be converted to variant calls. *Variant calling*, which is largely statistical inference, is the process of inferring a sample genome from the noisy short-read data. It relies on a reference genome, and its goal is to find variants between the sample and the reference; since humans share 99.9% of their DNA, the set of variants is small relative to the genome size. Here we focus on reference-based variant calling, rather than *de novo* assembly, which accounts for a large fraction of all variant calling algorithms.

Unfortunately, existing variant calling techniques are struggling to keep pace with the growing stream of sequence data, taking days to execute on even a single genome; this is a major obstacle in realizing the potential of this increasingly abundant wealth of genomic information. The computational burden of variant calling pipelines is in large part due to the genome's structure, namely regions of duplication, both near and exact [20]. These regions complicate the analysis because they increase uncertainty about how to process short-read data. Thus, some regions of the genome are harder to process than others.

We can leverage this knowledge to create a hybrid system for reducing the computational demands of variant calling, relying on an efficient tool wherever possible, and employing expensive and more accurate techniques in the more difficult regions characterized by duplication. In order to pursue this strategy, we must

identify the regions of the genome that can be effectively processed via existing techniques (*i.e.,* easier regions) vs. genome regions that require sophisticated tools (*i.e.,* harder regions). Previous work in evolutionary biology has made progress in characterizing the near duplication in the genome (*e.g.,* [19]); however, these efforts investigate duplicate regions with varying scales and complexities and therefore identify approximately half the genome as duplicates. This broad scope is not practical for our goal of improving variant calling efficiency, since we would like to identify harder regions that are limited to a small fraction of the genome to achieve computational gains from a hybrid system.

The three main contributions of our work are: **1) A novel partitioning strategy for identifying the easier and harder genomic regions.** We present a new paradigm for duplication in the genome, shaped by our focus on variant calling, in which we only consider genome substrings whose lengths match the short reads and which exhibit a high degree of similarity to at least one other substring. We call regions built from the overlap of these substrings *similar regions*.

2) An efficient algorithm to identify similar regions. The problem of detecting similar regions can be viewed as an unsupervised learning problem, in which we have a high-dimensional dataset consisting of three billion data points with a large (millions) but unknown number of clusters. Given the scale of our problem, pairwise comparison approaches are infeasible, and graph-based techniques are challenging since even representing the graph explicitly is difficult. We therefore present a distributed, MapReduce-style clustering approach based on union-find with algorithmic optimizations that can solve this problem.

3) A quantitative study of the impact of similar regions on variant calling using both synthetic and real datasets. We illustrate the effectiveness of embedding these similar regions into a hybrid variant calling approach. We first present results demonstrating the adverse impact that similar regions have on the first step of variant calling, short-read alignment. We next present extensive results on full genomes that clearly demonstrate that i) the similar regions account for a small fraction of the full genome (on the order of 7%), ii) standard variant callers struggle in the similar regions but perform quite well outside of these regions, and iii) the similar regions are small and fairly uniform in size, and can thus be naturally processed in parallel using more sophisticated and computationally expensive methods.

2 Related Work

In this section, we describe three categories of related genome partitioning strategies, which often result in the "blacklisting" of a particular genome subset. Moreover, we provide a quantitative comparison between various published genome blacklists and our proposed similar regions in Table 1.

The first class of genome partitions is based on the study of repeats in the human genome, often with the goal of understanding the evolutionary biology of the genome (*e.g.,* see [2], [8], [21], [3], [16]). One of the most well-known repeat databases is Repbase [7], an inclusive set of repeats with sequences that span a broad range of lengths and allow a significant degree of divergence among members of the same family. Indeed, Repbase[1] comprises 47% of the genome (see Table 1) and is thus ill-suited for a hybrid variant calling approach due to its large size.

The second class of partitions focus on the task of identifying regions of the genome that are particularly challenging for variant calling. However, these approaches all employ an orthogonal strategy of relying on the short reads themselves to identify these challenging regions. Many of these strategies are complementary to our approach, identifying regions enriched with sequencing errors, e.g., GC-rich regions, and/or sample-specific variation that could lead to low variant calling quality.

For instance, the accessible genome [1] aims to identify "accessible" regions for variant calling by aligning reads from many samples and developing heuristics primarily based on coverage to identify anomalous regions. The blacklist developed by the Pritchard Lab at Stanford[2] leverages a similar strategy to identify collapsed repeats that may hinder variant calling accuracy. The sizes of the inaccessible genome (!Accessible Genome) and the Pritchard blacklist (Pritchard) are given in Table 1.

Moreover, the ReduceReads module of GATK [5] compresses a BAM file on a per-sample basis, focusing on regions of the genome in which variants are concentrated. Read mapping quality [14], in which a MAPQ score

[1]We used the version of Repbase distributed via RepeatMasker open-3.3.0 - RepeatLibrary 20120124 (`http://www.repeatmasker.org/`)

[2]`http://eqtl.uchicago.edu/Masking/readme`

Blacklist	Overlap Size (%)	Total Size (%)
Repbase	169e6 (5.4%)	1.48e9 (47.3%)
!Accessible Genome	156e6 (5.0%)	472e6 (15.1%)
GEM Mappability	190e6 (6.1%)	266e6 (8.5%)
Pritchard	1.46e6 (0.046%)	6.47e6 (0.21%)

Table 1: Overlap between the similar regions and various blacklists, listed in descending order of size. Shown is both the size of the overlap and the size of the blacklist, in bp and as a fraction of the genome.

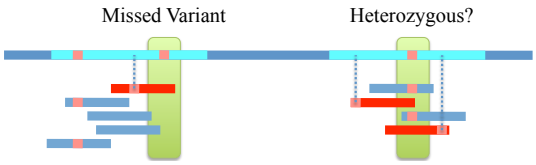


Figure 1: The problem of similarity. The top line represents the reference genome, and the cyan regions represent two similar regions. The red reads have been misaligned (they belong to the opposite similar region). Because of this misalignment, we will have trouble with variant calling. On the left, the red read could cause us to miss a variant; on the right, the two red reads might make us think that the sample is heterozygous at the pink location, whereas the sample is actually homozygous there.

is assigned to each read during the alignment phase to indicate aligner confidence, is another sample-specific metric related to our work. However, MAPQ scores are often poorly calibrated, and many practitioners do not trust it.

The GEM suite [6] comprises the third class of partition methods, and is the most closely related to our similar regions. GEM assesses the mappability of short reads by identifying how often each substring of length k occurs in the reference genome with up to m mismatches, and defining mappability as the inverse of this count. Hence, as in our work, GEM relies on the reference itself to determine the mappability of various regions, and the size of GEM blacklist is comparable to the similar regions, as noted in Table 1.[3] However, we provide more information than mere counts of a substring's neighbors; we provide the full set of connected components in the genome. Indeed, as discussed in Section 5, information about the structure of similar region families is crucial in order to disentangle the reads stemming from similar regions. Moreover, we build upon the preliminary studies in the GEM work, providing a thorough investigation of the impact of similar regions on variant calling and clearly demonstrating the need for alternative variant calling algorithms in these regions.

3 Methods

In our work, we consider reference-based variant calling approaches. This problem would be straightforward if the genome were a random string, since identical 100-base substrings within the human reference would occur with extremely low probability. Alas, the genome actually has both exact and near duplication. Exact duplicate regions are easy to locate, and if reads align against them, they may be flagged as impossible to uniquely align. However, as Figure 1 illustrates, near-duplicate regions are difficult not only to identify but also to process, since it is possible that many reads mapping to near-duplicate regions may be uniquely aligned to one of them. In what follows, we will discuss our solution for locating the near-duplicate, or *similar*, regions.

3.1 Detecting Similar Regions with SiRen

Our goal is to find similar regions whose characteristics are driven by the short reads themselves. To do so, we begin by creating a similarity graph to represent the genome. In our graph, there is a node for each overlapping substring of length R, where R is the length of the short reads of interest; we have approximately

[3] We used $k = 100$, $m = 1$, and disabled the approximation.

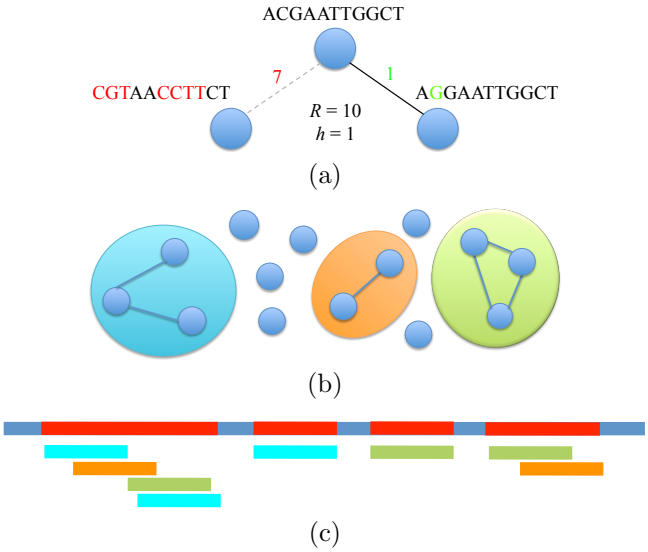


Figure 2: Schematic of SiRen algorithm. (a) We represent the genome as a graph, where each node is a length R substring of the reference and edges are based on the Hamming distance threshold h. (b) We locate the connected components in the genome graph. Singletons are discarded. (c) We merge the similar substrings to create *similar regions* (denoted by the red horizontal bars). Note that each of the similar regions is obtained by taking the overlap of similar substrings, colored according to their connected component.

three billion nodes. There is an edge between any two nodes for which the Hamming distance between the two substrings is less than or equal to a given threshold h. We then locate the connected components of this graph and discard any singletons.

Next, we identify similar regions from our similarity graph using the *SiRen* algorithm, which Figure 2 illustrates. We first map the substrings corresponding to nodes that belong to connected components back to the reference. If one or more of these substrings are found to overlap, we merge them. We refer to the resulting longer substring as a *similar region*; for clarity, we call a genome substring that does not contain any similar regions a *unique region*.

While this algorithm appears straightforward, naïve implementations, in particular for finding the edges of this graph, are intractable due to computation and storage costs. In the rest of this section, we present algorithms for improved efficiency that enable us to handle the scale of this problem (both in terms of computation and storage). The two main improvements are due to indexing and parallelization.

3.2 Index-Augmented Union-Find Algorithm

Given the nature of our problem, we identified the union-find algorithm as a promising approach. Union-find is a simple, well-known algorithm for finding the connected components in a graph [4]. Initially, each node is assigned to its own component, or cluster. Given a list of all the edges in the graph, we traverse each edge, and if the nodes connected by a given edge are not in the same cluster, we merge them.

Of course, we must first identify the edges in our graph before applying the union-find algorithm. A naïve approach is to compare each substring to each of the other three billion substrings. This comparison would be prohibitively expensive, however. Shrinking the runtime of this process is important to simplify development, parameter selection, and finding new similar regions whenever read lengths change or the reference is updated. Also, even if it were computationally feasible, we would perform many unnecessary comparisons since the adjacency matrix is likely to be sparse.

To avoid this wasted work, we build an index of the reference genome, where the key is a *seed* of length 20 bases, and the value is a list of locations in the reference at which that seed is found. We set the seed length equal to 20 because this value has been empirically determined to achieve a good tradeoff in the space of seed length versus number of hits; shorter seeds lead to excessive hits, while longer seeds lead to too few hits [22].

After building the index, we then scan over the substrings. For each substring s, we look up its non-overlapping seeds of length 20 in the index; since our read length is 100, we have 5 seeds. Each seed gives us a list of positions in the genome whose corresponding substrings are potentially similar to s. After looking up each of s's 5 seeds in the index, we obtain a list of all substrings potentially similar to s. The list length is orders of magnitude less than three billion, which is the number of comparisons we would have to perform under a naïve approach. Algorithm 1 describes this process.

Our approach for edge identification is a parametric one, where the parameter is the merge distance h. Moreover, since we require exact matches via seed lookups, our index-based algorithm may miss some potential edges if our merge threshold is greater than or equal to the number of seeds. In practice, we observe that a merge distance of 1 performed best in our experiments (see Section 4.3.3 for details), and thus we avoid the issue of missed edges.

Algorithm 1 Union-find with index optimization

R = read length, h = merge threshold, N = number of substrings in genome
Input: an index I, where the key is a 20-base seed, and the value is a list of all positions at which the seed occurs
Initialize: *parent*, an array of length N, where $parent(i) = i$ = cluster ID of the ith substring
for all genome substrings s of length R **do**
  for all seeds s' in s **do**
    $hits = I(s')$
    for all $s_h \in hits$ **do**
      if $Hamming(s_h, s) \leq h$ **then**
        Update $parent(s_h) = parent(s)$
      end if
    end for
  end for
end for
return *parent*

3.3 Efficient Parallel Implementation

Despite the dramatic reduction in comparisons, our index-based algorithm remains quite expensive given the scale of our data. Our goal is therefore to develop a parallel implementation of our approach. To facilitate parallelization, we use Spark, a MapReduce-like framework for distributed computing [23]. Moreover, we divide our index-based approach into subproblems, each of which solves an index-based union-find problem on a partition of the complete dataset, and merge the results of the subproblems on-the-fly as they complete.

The first step of our parallel, distributed algorithm involves partitioning the adjacency matrix so that each distributed task is allocated one partition. In our study, we use grid partitioning based on exploratory analysis with various partitioning schemes. Each partition has two ranges: one for the rows and one for the columns. The row range determines which positions should be included in the index as seeds. The column range determines which positions should be scanned as substrings. We scan each substring in the column range, and whenever we locate an edge (*i.e.,* we find that two substrings are similar), we merge the two substrings' clusters. Therefore, we are able to combine the edge location and merging into a single pass, avoiding the need to store the edges. This one-pass approach is important since this is already a memory-intensive process. Figure 3(b) illustrates this step. We also use the standard union-find optimizations of union by rank and path compression.

The cluster results from each partition are merged on the master node. Initially, the master node stores each substring as its own global cluster. Once a worker node computes the clusters for a single partition, the master updates the global clusters for the genome by merging any of the global clusters whose points appear together in the partition's clusters. This process is illustrated in Figure 3(c). Since each partition is independent, our approach is embarrassingly parallel and therefore exhibits *strong scaling*, since adding more machines to a fixed-size problem shrinks the runtime. See Algorithm 2 for a formal description of this process.

Algorithm 2 Parallelization of union-find

Initialize: $parent(i) = i$, where $length(parent) = N$
Partition the adjacency matrix according to a grid scheme, with dimension G
for all grid cells g with row range r and column range c **do**
  Initialize: $parent_g(i) = i$, where $length(parent) = |c|$
  Create a seed index I with the portion of the genome indicated by r
  for all genome substrings s beginning with positions in c **do**
    Find the neighbors of s using index I
    Update $parent_g$
  end for
  return $parent_g$
end for
if task corresponding to grid cell g completes **then**
  for all pairs (s_1, s_2) s.t. $parent_g(s_1) = parent_g(s_2)$ **do**
    if $parent(s_1) \neq parent(s_2)$ **then**
      Update $parent(s_1) = parent(s_2)$
    end if
  end for
end if

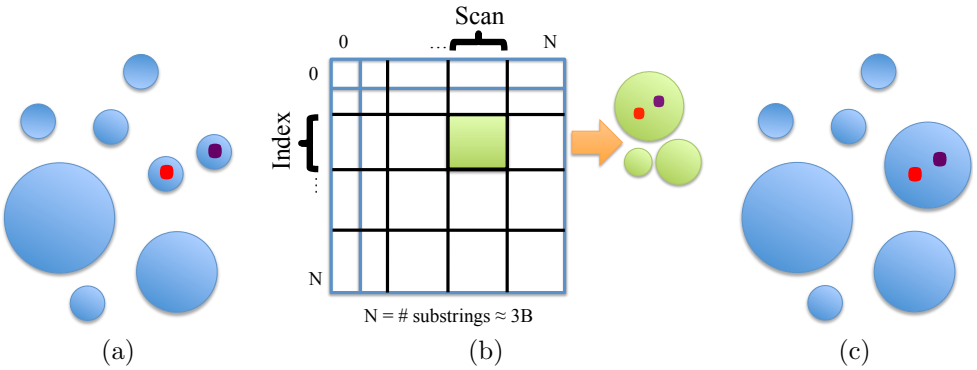


Figure 3: Parallelization of union-find. (a) In the global clusters, the red and purple points are initially in their own clusters. (b) To make union-find work at scale, we partition the genome in a grid fashion. Worker tasks process different partitions in parallel and identify the clusters within each partition. The green partition yields three clusters, with the red and purple points grouped together. (c) The master task then merges each partition's clusters into the global clusters. The global state is updated so that the red and purple points are in the same cluster.

4 Results

We begin by briefly explaining how we ran the similar region detection. Then, we provide three sets of experiments. First, we explore the relationship between alignment errors and reads that hit similar regions. Next, we provide an end-to-end analysis that looks at how the alignment errors impact the downstream stage of variant calling in similar regions. Third, we likewise perform an end-to-end analysis of variant calling with a mouse genome. Our goal for the three sets of experiments is both to show that similar regions affect the pipeline at each stage, as well as to show the robustness with which they detect an important signal in the data.

4.1 Similar Region Detection with SiRen

We ran SiRen on the human reference genome hg19, or GRCh37, which was released in February of 2009.[4] To do so, we used a node with a dual-socket Intel E5-2670 2.6 GHz processor with 16 cores and 128 GB of DDR3-1600MHz RAM. We allocated all 16 cores as well as 100 GB of memory to the tool. It takes about a week to run, given this setup; for smaller merge distances, it takes less time, and for longer merge distances, the runtime increases. For instance, with a merge distance of 4, the tool ran for 126 hours (approximately 5 days) of wall clock time, or 2016 CPU-hours. We note that without the optimizations described above, it was infeasible to run on the configuration described, since the memory demands were too great.

4.2 Impact on Alignment

We begin by focusing on alignment, which is the first step of most variant calling pipelines. Our goal is to use alignment to evaluate the similar regions that we find via SiRen. To do so, we have devised a method of generating labels for aligned data whereby we categorize each read on a spectrum from easy to hard. Our method is based on consensus across four alignment algorithms, or aligners, and yields labels with four gradations of easier or harder to align; *i.e.,* if all four aligners agree on a read's alignment location, we count it as easy, while if no aligners agree on a read's alignment location, we count it as hard. The other labels are obtained when two or three aligners agree. We selected this ensemble-style approach of label generation to avoid overfitting to any particular aligner; we picked these aligners because together, they are a good representation of the accuracy and efficiency spectrum.

Given a labeling scheme, we can now evaluate whether the regions we have obtained can explain our labeled data. We will judge the similar regions to be correlated with read difficulty if a high fraction of hard reads align to similar regions, while a low fraction of easy reads align to similar regions.

In what follows, we will show the correlation between aligner agreement and similar regions for both simulated and real data. The simulated data consists of 5 million pairs of reads generated from hg19 by the Mason simulator v.1.3.1.[5] The real data consists of 2 million pairs of reads sampled from the Illumina Platinum Genome Project's NA12878 genome.[6] The length of both the simulated and real reads is 100 bases. We aligned each dataset with four aligners: Bowtie2 v. 2.1.0 [10], BWA v. 0.7.4 [12], Novoalign v. 3.0.3, and SNAP v. 0.16alpha.6 [22].

We first describe how we evaluated the aligners on the simulated data. After aligning the reads with all four aligners, we assigned a label to each read based on the number of aligners that agreed on its location, where we determine agreement within a tolerance of 100 bases. We then segmented the reads according to their labels, and then for each label, we assessed how many of the reads hit similar regions. Figure 4(a) shows the results. We observe that almost all of the hard reads are in similar regions, while very few of the easy reads are in similar regions; therefore, we have achieved a close correlation between our labeled data and our similar regions. We provide these results for two merge distances, 1 and 10. We also obtained results for 2, 4, 6, and 8; as they follow the same pattern, we omit them here. Since more of the genome is contained in similar regions when the merge distance is large (see Figure 5), it is logical that more reads of each label would fall in clusters with a larger merge distance. However, since we still obtain a high fraction of the harder reads in clusters using 1 for the merge distance, while including much less of the genome in similar regions, we prefer 1. We provide further discussion of selecting the merge distance in Section 4.3.3.

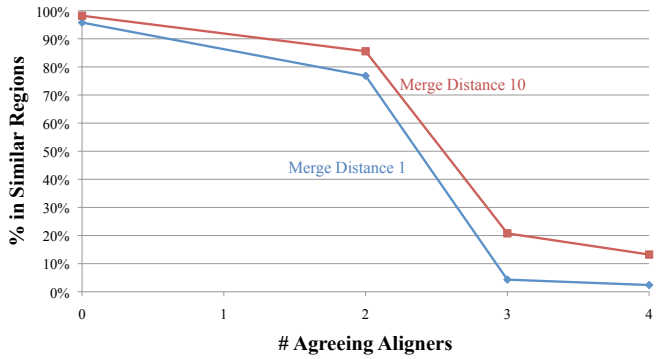
We carried out a similar analysis for the real data, to avoid overfitting to simulated data. The results, which are shown in Figure 4(b), are very similar to the results for simulated data. Again, reads that are hard to align are overwhelmingly in similar regions, while easy reads are not. We also observe the same trends regarding the merge distance.

These results demonstrate that one of the main reasons for alignment difficulty is the similar regions. We have shown this without overfitting to a single alignment algorithm or dataset; rather, we have shown that this holds for different aligners and for both simulated and real data.

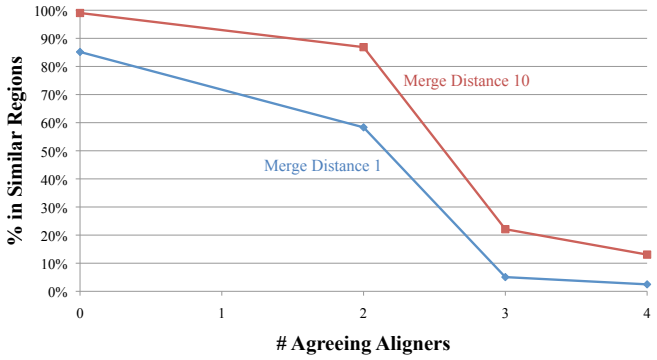
[4] `http://hgdownload.cse.ucsc.edu/downloads.html#human`

[5] `http://www.seqan.de/projects/mason/`

[6] `http://www.ebi.ac.uk/ena/data/view/ERP001229`



(a) Simulated



(b) Real

Figure 4: Aligner agreement, correlated with cluster membership. For both simulated (a) and real (b) data, hard reads coincide with high membership in similar regions, while easy reads fall in unique regions. These results are provided for two merge distances, 1 and 10, to illustrate that though a higher merge distance results in more hard reads appearing in similar regions, it also results in more easy reads being in similar regions. Therefore, we prefer a lower merge distance.

4.3 End-to-End Analysis: Variant Calling

We have shown that alignment, which is a key stage in the genomic processing pipeline, is strongly affected by the similar regions. However, this is an intermediate stage; ultimately, we care about variant calling, so in this section, we explore the impact of similar regions on variant calling accuracy. Our hypothesis is that due to alignment errors, variant calling will be more difficult in the similar regions than in the unique regions. In what follows, we validate this hypothesis using both simulated and real data.

4.3.1 Experimental Data

For this analysis, we used the SMaSH benchmarking suite [18]. SMaSH provides read datasets for both synthetic and real data, as well as labeled variants obtained from orthogonal sequencing technologies. Leveraging this labeled data, SMaSH also offers an evaluation tool to determine the accuracy, measured in both precision and recall, for variant calls made for the datasets. For both datasets, we consider both single nucleotide polymorphisms (SNPs) and short indels.

The synthetic human data from SMaSH is based on the variants detected in the Venter dataset [11]. These variants are inserted into the human reference genome hg19, and then simulated reads of length 100 bases with a coverage of 30× are produced via the simNGS tool [15]. Since all the variants are known, a comprehensive evaluation is possible; *i.e.,* we can report both precision and recall for the variant calls on this dataset.

Dataset	Tool	Variant	Merge Distance	Similar Regions		Unique Regions		Overall	
				Precision	Recall	Precision	Recall	Precision	Recall
Venter	mpileup	SNPs	1	93.8	78.6	99.0	98.5	98.7	97.0
			10	96.4	91.6	99.5	99.0		
		Indels	1	83.8	54.2	88.7	73.9	88.5	73.1
			10	86.8	64.1	89.4	78.6		
	GATK	SNPs	1	97.9	55.7	99.3	94.6	99.3	91.7
			10	98.4	80.0	99.5	96.0		
		Indels	1	86.1	67.5	91.6	90.5	91.5	89.5
			10	90.0	85.9	92.1	91.6		
NA12878	mpileup	SNPs	1	-	22.5	-	99.1	-	98.8
			10	-	82.8	-	99.3		
	GATK	SNPs	1	-	22.9	-	99.2	-	98.8
			10	-	83.0	-	99.4		

Table 2: Variant calling accuracy (in both precision and recall) for both simulated data & real data. Indels are omitted for NA12878 because SMaSH lacks indel validation data for this dataset.

The real human data from SMaSH is 64× coverage reads for the NA12878 sample. [7]. The reads are 101 bases long. Since this is real data, we do not know the types or locations of the true variants. However, SMaSH provides labeled variants from two SNP chips for this dataset. Note, however, that these labeled variants are not comprehensive; they are available only for the sections of the genome that the SNP chips were designed to interrogate. Therefore, we cannot determine the precision of variant calls, since precision depends on knowing the number of false positives. Hence, we report only recall for this dataset.

4.3.2 Experimental Setup

Our procedure for both datasets is as follows. Again using EC2, we begin by aligning each set of reads with BWA v. 0.6.1. We chose BWA due to its popularity among practitioners. Then, given the aligned reads, we produced variant calls with both mpileup (from SAMtools v. 0.1.19) [13] and GATK v. 2.6 [5]. We chose these tools because mpileup is known to be a very simple variant caller, while GATK is considered the state-of-the-art and is widely used by practitioners. We wanted to see how both a simple and a complicated tool would handle the similar regions.

Once we had obtained variant calls from both tools, we segmented all three variant call format (VCF) files for each dataset (one for the true variants, from SMaSH; one from mpileup; and one from GATK) according to which variants appeared in similar regions and which appeared in unique regions. Then, we used SMaSH's evaluation tool to determine the accuracy of the called variants both in and out of the similar regions.

4.3.3 Results

Table 2 displays the results. For the Venter dataset, which is synthetic, both mpileup and GATK perform worse in the similar regions than in the unique regions; this is more dramatic for the recall. For the NA12878 dataset, which is the real dataset, the difference in recall between the similar regions and the unique regions is much more pronounced. One likely cause is alignment errors, leading to reads being aligned to the wrong similar region, as Figure 1 depicts. Therefore, these results validate our hypothesis that variant calling is much harder in the similar regions than in the unique regions. These results are particularly interesting given that variant calling for SNPs and short indels is often regarded to be a solved problem, due to the high rates of overall precision and recall. However, we see that there is still a lot of work to do to achieve satisfactory accuracy in the similar regions. We suggest that developers of new variant callers concentrate

[7] It is available from http://smash.cs.berkeley.edu/datasets.html

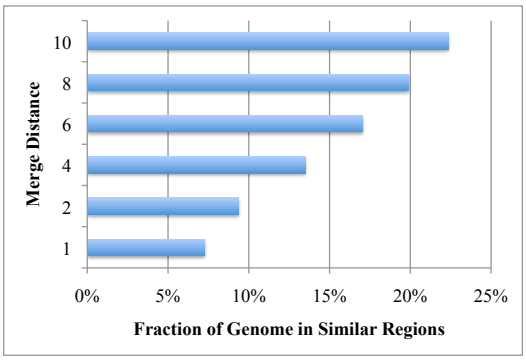


Figure 5: Percent of the genome in similar regions for different values of the merge distance. As we increase the merge distance, a greater fraction of the genome lies in similar regions.

on the similar regions rather than the unique regions, as the similar regions are clearly where the challenges lie.

Table 2 also gives us insight into how to choose the merge distance. The table presents results for merge distances 1 and 10; we also obtained results for merge distances 2, 4, 6, and 8 but omitted them for the sake of brevity. We observe that when we increase the merge distance from 1 to 10, the accuracy in the similar regions increases substantially, while the accuracy in the unique regions is barely impacted. The reason for this is that when you increase the merge distance and therefore also increase the fraction of the genome that lies in similar regions, you include some easier regions. Likewise, when you restrict the merge distance, you achieve a tighter correspondence between the hard regions and the similar regions. Therefore, we prefer a smaller merge distance, in this case 1, since we greatly reduce the size of the similar regions—from 22% at merge distance 10 to 7% at distance 1—without significantly impacting the performance of the variant calling algorithms in the unique regions. We will speak more about the desirability of limiting the size of the similar regions in Section 5.

Figure 6 illustrates a more subtle point. This figure compares, for our selected merge distance of 1, the fraction of variants (true, called by mpileup, and called by GATK) in similar regions. We observe that while Venter's true variants are distributed fairly evenly, NA12878 has few true variants in the similar regions. Since the Venter validation data is comprehensive while the NA12878 data is partial, we conjecture that the SNP chips used to obtain the NA12878 variants are biased against the similar regions, potentially because this data is more difficult to gather. Since SNP chips provide few variants in these regions, variant databases, which are largely based on SNP chips, likely also have relatively few variants in these regions. GATK, which is a much more sophisticated tool than mpileup, relies on assumptions about where variants are likely to appear based on databases of previously-detected variants. These factors may combine to make GATK discount SNPs that it identifies in these regions and may also explain why GATK underperforms mpileup on the Venter dataset, while it matches or exceeds mpileup's performance on the NA12878 dataset.

4.4 Variant Calling on Mouse Data

Since the SMaSH benchmark [18] also contains mouse data, we investigated whether our techniques would apply beyond human data. The mouse data has an attractive property in that it provides access to both real short-read data along with a comprehensive set of validated variants. Thus, we are able to state accuracy in both precision and recall.

To repeat the variant calling experiment with mouse data, we had to run the similar region detection tool on the mouse reference genome. As in Section 4.3, we tried a variety of merge distances to identify similar regions, aligned the mouse reads with BWA, and then ran mpileup and GATK on the aligned reads. We then segmented the true, mpileup, and GATK VCF files according to the similar regions.

Table 3 shows the results for merge distance 1. The results for the other merge distances, as in the case of the human data, showed improved performance in the similar regions without significant change in the unique regions. However, as this improved performance comes at the cost of increasing the percentage of

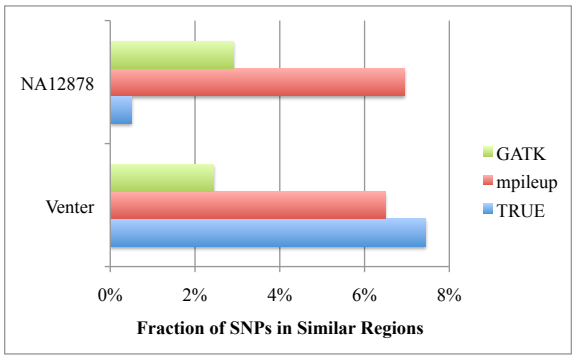


Figure 6: Fraction of SNPs in similar regions, for merge distance 1. For Venter, GATK vastly under-calls variants in similar regions. For NA12878, the true variants are underrepresented in similar regions, possibly due to bias in the chips made for detecting these variants. Since GATK relies heavily on prior knowledge about where variants are likely to be concentrated, the under-representation of variants in similar regions may explain its poor performance in those regions.

Tool	Variant Type	Similar Regions		Unique Regions		Overall		Runtime (h)
		Precision	Recall	Precision	Recall	Precision	Recall	
mpileup	SNPs	87.8	61.4	98.5	87.6	98.4	87.3	6
	Indels	60.4	39.3	83.2	81.3	83.1	81.1	
GATK	SNPs	80.9	67.7	98.0	95.2	97.8	94.9	107
	Indels	41.2	49.7	86.3	93.8	86.0	93.6	

Table 3: Variant calling accuracy (in both precision and recall) for mouse data, with merge distance 1.

the genome in similar regions, we again identify 1 as the optimal merge distance. Given our choice of merge distance, we observe that both GATK and mpileup perform worse, in both precision and recall, in the similar regions than in the unique regions. Therefore, we conclude that our techniques apply not only to the human genome but to other genomes as well.

5 Discussion

Recall that our overall goal is to find easier and harder regions so that we can employ a hybrid approach to short-read processing, where simple techniques are used on the easier regions and sophisticated techniques are used only on the harder regions. In this section, we reflect on how the similar regions we identified facilitate this goal.

A good easier/harder partitioning strategy will have two characteristics: first, we should identify a small portion of the genome as harder, and second, the harder regions should contain many short segments, so that we can launch parallel tasks, one per harder region, where no task is prohibitively expensive. To understand the first characteristic, consider that if we identify a large portion of the genome as harder, we have to run the sophisticated algorithm on a large portion of the genome, so we will not achieve much savings over running the sophisticated algorithm on the entire genome. The second characteristic is important because consider that if we identify just a few, long harder regions, it will limit the degree of parallelism that we can achieve.

Let us consider whether a partitioning scheme based on the similar regions achieves our two criteria. First, regarding the fraction of the genome in similar regions, we see in Figure 5 that with merge distance 1, which is our selected merge distance, only 7% of the genome lies in similar regions. Thus, with only 7% of the genome marked as harder, we will achieve an order of magnitude savings over running a sophisticated technique on the whole genome.

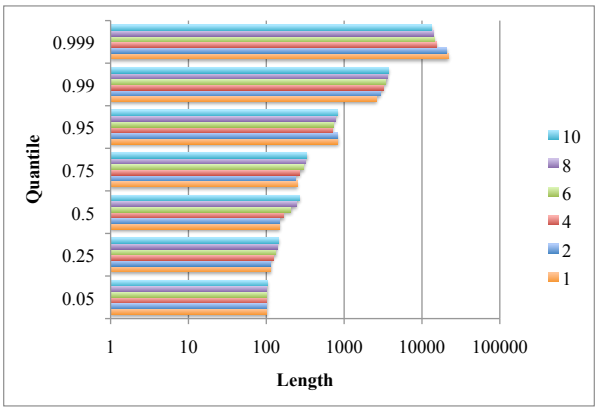


Figure 7: Quantiles of interest for lengths of similar regions, for different merge distances.

Second, regarding the size of the partitions, consider Figure 7. We observe that for our desired merge distance of 1, even at the 99th percentile, the similar regions are only a few thousand bases long. Figure 8 shows that at merge distance 1, there are approximately 600,000 of these partitions. Therefore, since we have identified many short partitions, we can obtain a high degree of parallelism.

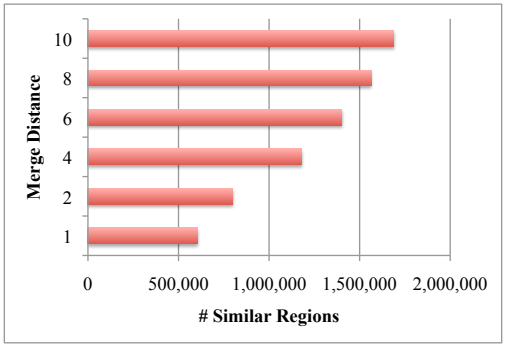


Figure 8: The number of similar regions, for different merge distances.

A straightforward technique for variant calling in the similar regions would be to approach each similar region in isolation. However, since similar regions are hotspots for alignment errors, we would likely suffer the same errors that mpileup and GATK make. Therefore, we are developing a strategy in which we consider similar region families (see Figure 9). In our graph representation, each node is a similar region, which is represented by a list of the components to which its substrings belong. Then, an edge connects any two nodes that share a component. We use a version of our union-find algorithm to find the connected components of this new graph; each component is a similar region family. An obvious concern with this strategy is that we will find large families for which disentangling the reads that originate from each region is too formidable a challenge. However, we have implemented this for chromosome 22 of the human reference genome, and we found that 81% of families are of size 2. Thus, we expect that when this is generalized to the whole genome, the vast majority of families will be suitable for disentangling.

6 Conclusions & Future Work

As the price of short-read sequencing continues to drop, the need to efficiently and accurately process the resulting genome data becomes ever greater. An intuitive approach to increasing the efficiency of variant calling without sacrificing the accuracy is to employ a hybrid scheme, in which simple and inexpensive techniques are applied to the easier genome regions, while expensive techniques are applied only where necessary. We have proposed that similar regions in the genome provide a natural basis for an easier/harder

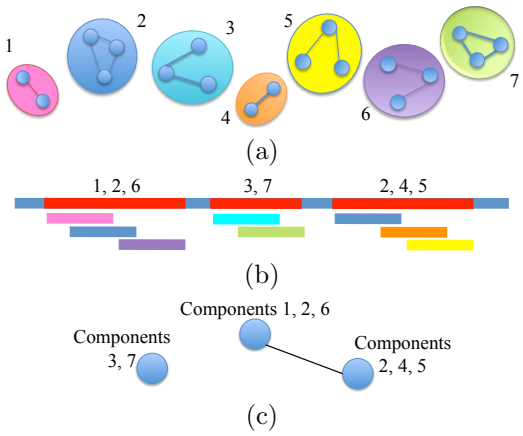


Figure 9: (a) When we locate the connected components in the substring graph, we give them each an ID. (b) Each similar region is made up of substrings from different components. (c) We group similar regions into families if they share at least one component.

partitioning scheme, and we have validated that the similar regions that we have identified correlate well with quantitative measures of difficulty in genome processing. We have also established that our similar regions possess characteristics that will facilitate parallelization in a hybrid approach.

Moving forward, we plan to design and build a hybrid variant calling pipeline, partitioning the genome based on the similar regions, and using sophisticated techniques to improve variant calling in the harder regions.

Acknowledgement

The authors thank Jesse Liptrap and Julie Newcomb for help with the variant calling experiments. They also thank the AMP-X team at UC Berkeley for helpful discussions and Beth Trushkowsky for feedback on the draft.

This research is supported in part by NSF CISE Expeditions Award CCF-1139158, LBNL Award 7076018, and DARPA XData Award FA8750-12-2-0331, and gifts from Amazon Web Services, Google, SAP, The Thomas and Stacey Siebel Foundation, Adatao, Adobe, Apple, Inc., Blue Goji, Bosch, C3Energy, Cisco, Cray, Cloudera, EMC2, Ericsson, Facebook, Guavus, HP, Huawei, Informatica, Intel, Microsoft, NetApp, Pivotal, Samsung, Schlumberger, Splunk, Virdata and VMware.

References

[1] G. R. Abecasis et al. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491:56–65, November 2012.

[2] Pankaj Agarwal and David J. States. The Repeat Pattern Toolkit (RPT): Analyzing the Structure and Evolution of the C. elegans Genome. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology (ISMB-94)*, pages 1–9. AAAI Press, 1994.

[3] Zhirong Bao and Sean R. Eddy. Automated De Novo Identification of Repeat Sequence Families in Sequenced Genomes. *Genome Research*, 12:1269–1276, 2002.

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 3 edition, July 2009.

[5] Mark A. DePristo et al. A framework for variation discovery and genotyping using next-generation dna sequencing data. *Nature Genetics*, 43(5), May 2011.

[6] Thomas Derrien, Jordi Estellé, Santiago Marco Sola, David G. Knowles, Emanuele Raineri, Roderic Guigó, and Paolo Ribeca. Fast Computation and Applications of Genome Mappability. *PLoS ONE*, 7(1), January 2012.

[7] J. Jurka, V. V. Kapitonov, A. Pavlicek, P. Klonowski, O. Kohany, and J. Walichiewicz. Repbase Update, a database of eukaryotic repetitive elements. *Cytogenic and Genome Research*, 2005.

[8] Stefan Kurtz and Chris Schleiermacher. REPuter: fast computation of maximal repeats in complete genomes. *Bioinformatics*, 15(5):426–427, February 1999.

[9] Eric S. Lander et al. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, February 2001.

[10] Ben Langmead and Steven L. Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature Methods*, 9(4):357–359, March 2012.

[11] S. Levy, G. Sutton, P.C. Ng, L. Feuk, and A.L. Halpern et al. The Diploid Genome Sequence of an Individual Human. *PLoS Biology*, 5(10), September 2007.

[12] Heng Li and Richard Durbin. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25(14):1754–1760, May 2009.

[13] Heng Li, Bob Handsaker, Alex Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, and 1000 Genome Project Data Processing Subgroup. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16):2078–2079, August 2009.

[14] Heng Li, Jue Ruan, and Richard Durbin. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Research*, 18:1851–1858, 2008.

[15] Tim Massingham. simNGS and simLibrary software for simulating next-gen sequencing data, 2012.

[16] Alkes L. Price, Neil C. Jones, and Pavel A. Pevzner. De novo identification of repeat families in large genomes. *Bioinformatics*, 21:i351–i358, March 2005.

[17] S. Schuster. Next-generation sequencing transforms today's biology. *Nature Methods*, 5:16–18, 2008.

[18] Ameet Talwalkar, Jesse Liptrap, Julie Newcomb, Christopher Hartl, Jonathan Terhorst, Kristal Curtis, Ma'ayan Bresler, Yun Song, Michael I. Jordan, and David Patterson. SMaSH: A Benchmarking Toolkit for Human Genome Variant Calling. *arXiv:1310.8420*, October 2013.

[19] Maja Tarailo-Graovac and Nansheng Chen. Using RepeatMasker to identify repetitive elements in genomic sequences. *Current Protocols in Bioinformatics*, 25:4.10.1–4.10.14, March 2009.

[20] Todd J. Treangen and Steven L. Salzberg. Repetitive DNA and next-generation sequencing: computational challenges and solutions. *Nature*, 13, January 2012.

[21] Natalia Volfovsky, Brian J. Haas, and Steven L. Salzberg. A clustering method for repeat analysis in DNA sequences. *Genome Biology*, 2(8), June 2001.

[22] Matei Zaharia, William J. Bolosky, Kristal Curtis, Armando Fox, David Patterson, Scott Shenker, Ion Stoica, Richard M. Karp, and Taylor Sittler. Faster and More Accurate Sequence Alignment with SNAP. November 2011.

[23] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. In *NSDI 2012*, April 2012.